ESTRUCTURA DE DATOS 1
Código ST0245

Taller en Sala 3
Recursión Avanzada

Objetivo: Identificar el caso base y el caso general de un problema definido recursivamente.

Consideraciones: Lean y verifiquen las consideraciones de entrega,

Trabajo en Parejas

Mañana, plazo de entrega

Docente entrega plantilla de código en GitHub

Sí .cpp, .py o .java

No .zip, .txt, html o .doc

Alumnos entregan código sin comprimir GitHub

En la carpeta Github del curso, hay un código iniciado y un código de pruebas (tests) que pueden explorar para solucionar los ejercicios

Estructura del documento: **a)** Datos de *vida real*, **b)** I*ntroducción* a un problema, **c)** Problema a resolver, **d)** Ayudas. Identifiquen esos elementos así:

a) **b)** **c)** **d)**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicios a resolver

En la vida real, las torres de Hanoi se utilizan para definir esquemas de rotación de backups http://bit.ly/2hAqRkX. También se utilizan como un examen neurológico para evaluar deficiencias en el lóbulo frontal http://bit.ly/2fYZmRL

1 En la película “El planeta de los simios: R(evolución)” (2011), las torres de Hanoi se utilizaron para evaluar la inteligencia de los simios.

En la vida real, las torres de Hanoi se han utilizado en neuropsicología para evaluar niños con trastorno obsesivo compulsivo. Un problema que tienen los psicólogos cuando hacen estos exámenes, es que necesitan tener a la mano la solución.

Implementen un algoritmo recursivo que muestre en la pantalla la solución al problema de las torres de Hanoi para un número de discos ingresado por el usuario.

2 Durante el desarrollo de la Operación Fénix, por parte de las Fuerzas Militares de Colombia, ocurrida en marzo 1 de 2008, fueron encontrados varios computadores de Raúl Reyes que detallaban información histórica de las actividades de las FARC.

Los computadores fueron enviados a la Interpol para analizar la información que se encontró y la veracidad de la misma. La información se encontraba encriptada.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Una solución para desencriptar un archivo es realizar un ataque de fuerza bruta. Este ataque consiste en probar todas las permutaciones posibles hasta encontrar la contraseña que permite el acceso.

Implemente un algoritmo recursivo para desencriptar un archivo, probando todas las permutaciones posibles de los caracteres de una cadena de caracteres, teniendo en cuenta, por simplicidad, que las letras de la contraseña no se repiten y que una la contraseña es una permutación de la cadena de caracteres ingresada por el usuario.

Utilicen el algoritmo que genera las permutaciones para desencriptar el archivo *archivoEncriptado.txt* que se encuentra en GitHub. Utilicen la clase *AdvancedEncryptionStandard*. El *password* es una permutación de "abcd".

3 **[Ejercicio Opcional]** En 1996, el computador *Deep Blue*, desarrollado por IBM, venció al campeón mundial de ajedrez Garry Kasparov.

Posteriormente, en 2016, el programa *AlphaGo*, desarrollado por *Google*, venció a uno de los mejores jugadores de *Go* en el mundo.

Programas que sean capaz de jugar ajedrez o *Go*, son de mucho interés para las grandes multinacionales porque dan los cimientos para sistemas de computación cognitiva como *Watson* de *IBM*. Un primer paso para construir sistemas que jueguen ajedrez, es resolver el acertijo de las *n reinas*. La importancia de este problema también es de interés para las universidades; de hecho, la universidad de *Saint Andrews*, en Escocia, ofrece 1 millón de dólares a quien resuelva este problema para tableros de 1000x1000, es decir, para n = 1000. Ver más en: https://www.clarin.com/sociedad/ofrecen-millon-dolares-resuelva-problema-ajedrez-ahora-indescifrable_0_rJ2YdZk9-.html

Implementen un algoritmo que resuelva el problema de las *n reinas*. Este problema consiste en ubicar en un tablero de ajedrez de *n* por *n* a n reinas de tal forma que no se ataquen entre sí.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

[Ejercicio Opcional] Implementen la funcionalidad del tarrito de Paint, representeando su lienzo como una matriz de enteros. El método recibe la matriz, la posición en la que quieren pintar *(x,y)* y el color (un número).

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para resolver los Ejercicios

Ejercicio 1

Pista 1: El problema de las torres de Hanoi es el siguiente: Tenemos 3 postes y *n* discos que colocar en los postes. Los discos difieren en tamaño e inicialmente están en uno de los postes, en orden del más grande (disco *n*) al más pequeño (disco 1).

La tarea es mover todos los discos a otro poste, pero obedeciendo las siguientes reglas:

- Sólo se puede mover un disco a la vez
- Nunca coloquen un disco sobre uno más pequeño que él

Implementen un método que imprima en la pantalla los movimientos que hay que hacer para mover los discos de un poste a otro. Por ejemplo, para n = 2:

- **Mover un disco de la torre 1 a la 2**
- **Mover un disco de la torre 1 a la 3**
- **Mover un disco de la torre 2 a la 3**

Pista 2: Definan la función de esta forma:

```
private static void torresDeHannoiAux(int n, int origen, int
intermedio, int destino) {

}

public static void torresDeHannoi(int n) {
    torresDeHannoiAux(n, 1, 2, 3);
}
```

Pista 3: https://www.youtube.com/watch?v=5_6nsViVM00

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicio 2

Pista 1: Escriban un programa que calcula las n! permutaciones de una cadena. Como un ejemplo, cuando uno le dicen “abc” debe dar la siguiente salida:

bca cba cab acb bac abc

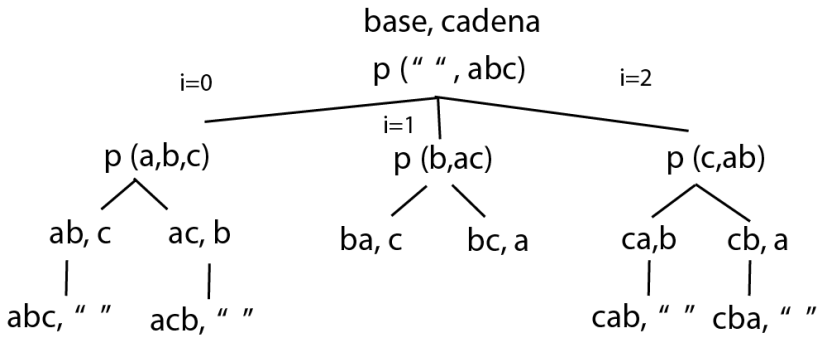
Pista 2: Consideren el siguiente código:

```
private static void permutationsAux(String base, String s) {
   …
}

public static void permutations(String s) {
   permutations(“”, s);
}
```

Pista 3: No se preocupen en qué orden obtiene las permutaciones.

Pista 4: La función recursiva debe generar el siguiente árbol de ejecución para generar las permutaciones de longitud 3 de la cadena “abc”.



PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicio 3

Ejemplo 1: Estas son las 2 soluciones para el problema de las n reinas con n = 4

#	#	Q	#
Q	#	#	#
#	#	#	Q
#	Q	#	#

#	Q	#	#
#	#	#	Q
Q	#	#	#
#	#	Q	#

Pista 1:
Guía para la Implementación

1. Escriban un método para verificar si puede poner una reina en determinada posición. Recuerden que debido a la forma como representamos los tableros es imposible tener dos reinas en una misma fila, por lo que no necesitan verificar esta condición. El parámetro *r* hace referencia a la fila, y el *c* a la columna.

```
private static boolean puedoPonerReina(int r, int c, int[]
tablero) {
          // complete...
}
```

2. Implementen el método con recursión. El parámetro *r* representa la fila, el parámetro *n* el número de reinas. Esta función retorna un entero que representa el número de soluciones que existen para las n-reinas. Por ejemplo, para n=4, retorna 2 y para n=8 retorna 92.

```
private static int nReinas(int r, int n, int[] tablero) {
// complete...
}
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Se itera fila por fila (hay *n* filas) y se van poniendo reinas si es válido hacerlo.

3. Llamen el método creado en el paso **2** desde el *wrapper*.

```
public static int nReinas(int n) {
     // complete...
}
```

Ejercicio 4

Pista 1: El tarrito de Paint sigue un algoritmo llamado *Flood Fill* https://bit.ly/2Ke9k3a La idea es que el lienzo está representado con una matriz de enteros.

Supongamos que dentro del lienzo está dibujado un polígono relleno de color 2 y el resto está de color 0. Entonces si uno le dice que quiere pintar una posición dentro del polígono relleno de color 2, con el color 5, todo ese polígono debe quedar de color 5. Es como si el color se expandiera como un virus, hacia arriba, hacia abajo, hacia los lados... algo así:

```
00000          00000
00222          00555
02222 --->     05555
00222          00555
```

Para efectos del taller, realicen una implementación de este algoritmo usando recursión. Una vez funcione el algoritmo, se sugiere leer más en Wikipedia porque explican mejoras al algoritmo y otras formas de implementarlo.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 Ext. 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agenden una cita dando clic en la pestaña

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473